1818 Library ◆ Suite 600 ◆ Reston, VA 20190-5602 ◆ (703) 326-2861  
<http://www.metsci.com>



Planner Issues   
V1.1

August 5, 2015

Thomas M. Kratzke

Sr. Analyst

Contents

[1. Introduction 1](#_Toc426535371)

[2. Optimize on Search Patterns rather than Rectangles 2](#_Toc426535372)

[3. Don’t use All Available Search Effort. 3](#_Toc426535373)

[4. Use a Constant Search Pattern Buffer, rather than ½ track-spacing 4](#_Toc426535374)

[5. Correct the Squatter’s Rights Problem. 5](#_Toc426535375)

[6. Creep nearby SRUs in the Same Direction. 6](#_Toc426535376)

List of Algorithms

**No table of figures entries found.**

List of Code

**No table of figures entries found.**

List of Figures

**No table of figures entries found.**

List of Equations

**No table of figures entries found.**List of Snippets

**No table of figures entries found.**

List of Tables

**No table of figures entries found.**

Document Revision History

|  |  |
| --- | --- |
| Version Date | Comments |
| Apr 08, 2015 | For *Sim* V1.5 |
|  |  |

# Introduction

The current *Planner* algorithm works well enough for basic cases, but there are many complexities that can and do arise, that can lead Planner to create poor solutions. In this note, we will discuss modifications that now appear to have a high probability of being useful. Prototypes for some of these algorithms exist in a separate branch of code, and this code has demonstrated that it does better than SAROPS’ planner for a particular complex test case.

The enhancements that we would work on are:

1. Optimize on Search Patterns rather than Rectangles.
2. Don’t require the use of all available search area.
3. Use a fixed buffer between patterns, rather than ½ a track-spacing.
4. Correct the algorithm’s tendency to invoke “squatter’s rights.”
5. For nearby SRUs, give them parallel search areas and have them creep in the same direction.

# Optimize on Search Patterns rather than Rectangles

Currently, Planner assigns an SRU a box, specified by:

1. The center point (Lat/Lon),
2. The length,
3. The width,
4. The orientation, and
5. Whether or not it is a PS or a CS pattern.

The pattern is then derived from this box, and it is assumed that the box will use all of the SRU’s available search effort.

It would be better if Planner specified an SRU’s assignment by giving it:

1. Commence Search Point (Lat/Lon),
2. The length of the first (search) leg
3. The direction of the first leg, and
4. The length of the second (cross) leg. If this second leg’s length is negative, that would indicate first-turn-left. Positive indicates first-turn-right.

This would provide a more flexible framework for specifying ladder patterns because the last leg would not need to be complete. The corresponding exclusion polygon, i.e., the polygon that may not intersect other SRUs’ polygons, could easily be derived, especially if we use a fixed buffer (see §4 below).

In short, the pattern has a more direct effect on the optimization than the rectangle, and it would be more straightforward to focus on the pattern.

By focusing on the pattern rather than the exclusion area, it would be simpler to introduce constraints on the pattern’s characteristics, and algorithms for improving the patterns. For example, if a rectangle is specified using the current Planner method, the length of the cross leg is simply a by-product of the rectangle’s specification and not a parameter. The current Planner can and does constrain the cross leg to be above a minimum, but in this new formulation, we would simply say that the variable representing the cross leg be above a given minimum.

Another example of the advantage of focusing on patterns, is that the derivation of the actual pattern could follow other rules. In particular, instead of flying legs into and past the legs’ landfall, the search leg could be shortened. Then the pattern would continue until it simply uses up its allocated length.

There appears to be a tradeoff; the exclusion area is now the entity that has to be derived. However this is not very onerous for the types of patterns being discussed. In fact, we have expanded the set of patterns for a given search length from pure complete ladder patterns to ladder patterns with the last leg shorter. This produces an exclusion area that is a pentagon, but still convex. This area is easily derived, and clearing overlap violations for arbitrary pairs of convex polygons is no more difficult than for rectangles. Part of this prototype work has been to develop more efficient algorithms for such clearing.

# Don’t use All Available Search Effort.

Going hand-in-hand with §2is the ability to use less than the maximum search area. Because of exclusion areas, the requirement that we use the entire available search length often leads to problems. For example, if a relatively ineffective SRU with a large search area is placed over a high probability area, then no other SRU can come close. If instead of requiring that the entire search area be used, we simply constrained the amount of search area to be at most the amount available, this SRU could be curtailed to allow room for other SRUs.

In addition to getting better answers, it could get more “attractive answers.” Often an SRU will fly vast stretches over land simply to use up its available effort. By allowing Planner to curtail the amount of effort used, these useless legs could be eliminated.

This would force us to enlarge an SRU’s solution to include “amount used.” There are mathematical advantages to this. The formulation mentioned in §2 would eliminate discreet variables and open up non-linear optimization techniques. We would simply add the “amount used” variable (and possibly penalize solutions very slightly for using a lot of search effort). There are public-domain libraries that purport to handle non-linear optimization problems of roughly this size, in particular, NlOpt from MIT should be able to handle up to five SRUs (30 variables). It remains to be seen how effective this would be, but it does not appear to be hard to use them.

# Use a Constant Search Pattern Buffer, rather than ½ track-spacing

Currently, Planner separates SRUs’ paths according to their track spacing. It would be a very straightforward modification to the way Planner creates an exclusion area from the formulation of §2, to implement this. It would be easiest if there were one buffer width for all SRUs, and each SRU’s exclusion polygon would be based on ½ of this value.

It really seems to make more sense to have a fixed buffer. In fact, the current method decreases the buffer the tighter the turns are, and that seems counter-intuitive.

# Correct the Squatter’s Rights Problem.

This problem is that Planner often has trouble “dislodging” one SRU even though a second SRU should at least share that position or even replace the first SRU.

One attack on this problem would be to allow the 1st SRU’s effort to be decreased so that its search area would shrink and the 2nd SRU’s area could gain ground. This is discussed in §3.

Another approach has been prototyped, and it also addresses the issue of §6. The current Planner works hard to prevent large overlap when laying down an initial guess for the SRUs’ positions.

A prototype exists that uses many of the ideas in this note, and this prototype does *not* focus on herculean avoidance overlap. In this approach, after all of the SRUs have been placed, there will be clusters of SRUs that overlap. These SRUs are then simultaneously re-placed to not overlap, and to cover the region that they were assigned. This “re-placing” is done so that the search legs are parallel and the creep of the SRUs’ solutions is in the same direction.

Using this approach, the “re-placing” algorithm could be implemented to curtail the “gluttony” of any particular SRU.

# Creep nearby SRUs in the Same Direction.

This is part of the algorithm introduced in §5. When Planner recognized a cluster, a set of angles is tried for the common angle of the SRUs’ first leg, and the region is divided across the extent of the region. The direction of the first leg will be the same or 180 off, and, then it’s straightforward to enforce a common direction of creep.